Universidade do Minho
Escola de Engenharia

Cálculo de Programas

Trabalho Prático (2023/24)

Lic. em Ciências da Computação

Grupo G99

axxxxxx Nome
axxxxxx Nome
axxxxxx Nome

abril 2024

Preâmbulo

Cálculo de Programas tem como objectivo principal ensinar a programação de computadores como uma disciplina científica. Para isso parte-se de um repertório de *combinadores* que formam uma álgebra da programação e usam-se esses combinadores para construir programas *composicionalmente*, isto é, agregando programas já existentes.

Na sequência pedagógica dos planos de estudo dos cursos que têm esta disciplina, opta-se pela aplicação deste método à programação em Haskell, sem prejuízo da sua aplicação a outras linguagens funcionais. Assim, o presente trabalho prático coloca os alunos perante problemas concretos que deverão ser implementados em Haskell. Há ainda um outro objectivo: o de ensinar a documentar programas, a validá-los e a produzir textos técnico-científicos de qualidade.

Antes de abordarem os problemas propostos no trabalho, os grupos devem ler com atenção o anexo A onde encontrarão as instruções relativas ao sofware a instalar, etc.

> Valoriza-se a escrita de *pouco* código, que corresponda a soluções simples e elegantes mediante a utilização dos combinadores de ordem superior estudados na disciplina.
> Recomenda-se ainda que o código venha acompanhado de uma descrição de como funciona e foi concebido, apoiado em diagramas explicativos. Para instruções sobre como produzir esses diagramas e exprimir raciocínios de cálculo, ver o anexo D.

Problema 1

No passado dia 10 de Março o país foi a eleições para a Assembleia da República. A lei eleitoral portuguesa segue, como as de muitos outros países, o chamado Método de Hondt para selecionar os candidatos dos vários partidos, conforme os votos que receberam. E, tal como em anos anteriores, há sempre notícias a referir a quantidade de votos desperdiçados por este método. Como e porque é que isso acontece?

Pretende-se nesta questão construir em Hakell um programa que implemente o método de Hondt. A Comissão Nacional de Eleições descreve esse método nesta página, que deverá ser estudada para resolver esta questão. O quadro que aí aparece,

Divisor	Partido			
	A	B	C	D
1	**12000**	**7500**	**4500**	**3000**
2	**6000**	**3750**	2250	1500
3	**4000**	2500	1500	1000
4	3000	1875	1125	750

mostra o exemplo de um círculo eleitoral que tem direito a eleger 7 deputados e onde concorrem às eleições quatro partidos A, B, C e D, cf:

$$\mathbf{data}\ Party = A \mid B \mid C \mid D\ \mathbf{deriving}\ (Eq, Ord, Show)$$

A votação nesse círculo foi

$$[(A, 12000), (B, 7500), (C, 4500), (D, 3000)]$$

sendo o resultado eleitoral

$$result = [(A,3),(B,2),(C,1),(D,1)]$$

apurado correndo

$$result = final\ history$$

que corresponde à última etapa da iteração:

$$history = [\text{for } step\ db\ i \mid i \leftarrow [0..7]]$$

Verifica-se que, de um total de 27000 votos, foram desperdiçados:

$$wasted = 9250$$

Completem no anexo F as funções que se encontram aí indefinidas[1], podendo adicionar funções auxiliares que sejam convenientes. No anexo E é dado algum código preliminar.

Problema 2

A biblioteca *LTree* inclui o algoritmo "mergesort" (*mSort*), que é um hilomorfismo baseado função

$$merge :: Ord\ a \Rightarrow ([a],[a]) \to [a]$$

que junta duas listas previamente ordenadas numa única lista ordenada.

Nesta questão pretendemos generalizar *merge* a k-listas (ordenadas), para qualquer k finito:

$$mergek :: Ord\ a \Rightarrow [[a]] \to [a]$$

Esta função deverá ser codificada como um hilomorfismo, a saber:

$$mergek = [\![f, g]\!]$$

1. Programe os genes f e g do hilomorfismo *mergek*.
2. Estenda *mSort* a

   $$mSortk :: Ord\ a \Rightarrow Int \to [a] \to [a]$$

   por forma a este hilomorfismo utilizar *mergek* em lugar de *merge* na estapa de "conquista". O que se espera de *mSortk k* é que faça a partição da lista de entrada em k sublistas, sempre que isso for possível. (Que vantagens vê nesta nova versão?)

Problema 3

A fornecer na segunda edição deste enunciado

Problema 4

A fornecer na segunda edição deste enunciado

[1] Cf. $\perp$ no código.

Anexos

A Natureza do trabalho a realizar

Este trabalho teórico-prático deve ser realizado por grupos de 3 alunos. Os detalhes da avaliação (datas para submissão do relatório e sua defesa oral) são os que forem publicados na página da disciplina na *internet*.

Recomenda-se uma abordagem participativa dos membros do grupo em **todos** os exercícios do trabalho, para assim poderem responder a qualquer questão colocada na *defesa oral* do relatório.

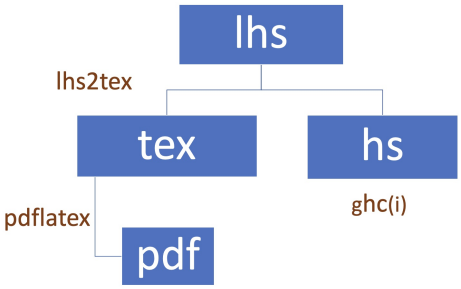
Para cumprir de forma integrada os objectivos do trabalho vamos recorrer a uma técnica de programação dita "literária" [1], cujo princípio base é o seguinte:

> *Um programa e a sua documentação devem coincidir.*

Por outras palavras, o **código fonte** e a **documentação** de um programa deverão estar no mesmo ficheiro.

O ficheiro `cp2324t.pdf` que está a ler é já um exemplo de programação literária: foi gerado a partir do texto fonte `cp2324t.lhs`[1] que encontrará no material pedagógico desta disciplina descompactando o ficheiro `cp2324t.zip`.

Como se mostra no esquema abaixo, de um único ficheiro (*lhs*) gera-se um PDF ou faz-se a interpretação do código Haskell que ele inclui:



Vê-se assim que, para além do GHCi, serão necessários os executáveis pdflatex e lhs2TeX. Para facilitar a instalação e evitar problemas de versões e conflitos com sistemas operativos, é recomendado o uso do Docker tal como a seguir se descreve.

B Docker

Recomenda-se o uso do container cuja imagem é gerada pelo Docker a partir do ficheiro `Dockerfile` que se encontra na diretoria que resulta de descompactar `cp2324t.zip`. Este container deverá ser usado na execução do GHCi e dos comandos relativos ao LaTeX. (Ver também a `Makefile` que é disponibilizada.)

Após instalar o Docker e descarregar o referido zip com o código fonte do trabalho, basta executar os seguintes comandos:

[1] O sufixo 'lhs' quer dizer *literate Haskell*.

```
$ docker build -t cp2324t .
$ docker run -v ${PWD}:/cp2324t -it cp2324t
```

NB: O objetivo é que o container seja usado *apenas* para executar o GHCi e os comandos relativos ao LaTeX. Deste modo, é criado um *volume* (cf. a opção `-v ${PWD}:/cp2324t`) que permite que a diretoria em que se encontra na sua máquina local e a diretoria `/cp2324t` no container sejam partilhadas.

O grupo deverá visualizar/editar os ficheiros numa máquina local e compilá-los no container, executando:

```
$ lhs2TeX cp2324t.lhs > cp2324t.tex
$ pdflatex cp2324t
```

lhs2TeX é o pre-processador que faz "pretty printing" de código Haskell em LaTeX e que faz parte já do container. Alternativamente, basta executar

```
$ make
```

para obter o mesmo efeito que acima.

Por outro lado, o mesmo ficheiro `cp2324t.lhs` é executável e contém o "kit" básico, escrito em Haskell, para realizar o trabalho. Basta executar

```
$ ghci cp2324t.lhs
```

O grupo deve abrir o ficheiro `cp2324t.lhs` num editor da sua preferência e verificar que assim é: todo o texto que se encontra dentro do ambiente

```
\begin{code}
...
\end{code}
```

é seleccionado pelo GHCi para ser executado.

C Em que consiste o TP

Em que consiste, então, o *relatório* a que se referiu acima? É a edição do texto que está a ser lido, preenchendo o anexo F com as respostas. O relatório deverá conter ainda a identificação dos membros do grupo de trabalho, no local respectivo da folha de rosto.

Para gerar o PDF integral do relatório deve-se ainda correr os comando seguintes, que actualizam a bibliografia (com BibTeX) e o índice remissivo (com `makeindex`),

```
$ bibtex cp2324t.aux
$ makeindex cp2324t.idx
```

e recompilar o texto como acima se indicou. (Como já se disse, pode fazê-lo correndo simplesmente `make` no container.)

No anexo E disponibiliza-se algum código Haskell relativo aos problemas que são colocados. Esse anexo deverá ser consultado e analisado à medida que isso for necessário.

Deve ser feito uso da programação literária para documentar bem o código que se desenvolver, em particular fazendo diagramas explicativos do que foi feito e tal como se explica no anexo D.

> **Importante:** o grupo deve evitar trabalhar fora deste ficheiro lhs que lhe é fornecido. Se, para efeitos de divisão de trabalho, o decidir fazer, deve **regularmente integrar** e validar as soluções que forem sendo obtidas neste lhs, garantindo atempadamente a compatibilidade com este. Se não o fizer corre o risco de vir a submeter um ficheiro que não corre no GHCi e/ou apresenta erros na geração do PDF.

D Como exprimir cálculos e diagramas em LaTeX/lhs2TeX

Como primeiro exemplo, estudar o texto fonte (lhs) do que está a ler[1] onde se obtém o efeito seguinte:[2]

$$
\begin{array}{ll}
& id = \langle f, g \rangle \\
\equiv & \quad \{\ \text{universal property}\ \} \\
& \left\{ \begin{array}{l} \pi_1 \cdot id = f \\ \pi_2 \cdot id = g \end{array} \right. \\
\equiv & \quad \{\ \text{identity}\ \} \\
& \left\{ \begin{array}{l} \pi_1 = f \\ \pi_2 = g \end{array} \right. \\
\square &
\end{array}
$$

Os diagramas podem ser produzidos recorrendo à *package* LaTeX xymatrix, por exemplo:

$$
\begin{array}{ccc}
\mathbb{N}_0 & \xleftarrow{\text{in}} & 1 + \mathbb{N}_0 \\
{\scriptstyle (\!|g|\!)}\downarrow & & \downarrow{\scriptstyle id + (\!|g|\!)} \\
B & \xleftarrow[g]{} & 1 + B
\end{array}
$$

E Código fornecido

Problema 1

Tipos básicos:

$$
\begin{array}{l}
\textbf{type}\ Votes = \mathbb{Z} \\
\textbf{type}\ Deputies = \mathbb{Z}
\end{array}
$$

Dados:

$$
\begin{array}{l}
db :: [(Party, (Votes, Deputies))] \\
db = \text{map}\ \ f\ vote\ \textbf{where}\ f\ (a, b) = (a, (b, 0)) \\
vote = [(A, 12000), (B, 7500), (C, 4500), (D, 3000)]
\end{array}
$$

Apuramento:

[1] Procure e.g. por `"sec:diagramas"`.

[2] Exemplos tirados de [2].

$$final = \text{map}\ \ (id \times \pi_2) \cdot last$$
$$total = sum\ (\text{map}\ \ \pi_2\ vote)$$
$$wasted = waste\ history$$

F Soluções dos alunos

Os grupos devem colocar neste anexo as suas soluções para os exercícios propostos, de acordo com o "layout" que se fornece. Não podem ser alterados os nomes ou tipos das funções dadas, mas pode ser adicionado texto, diagramas e/ou outras funções auxiliares que sejam necessárias.

> **Importante**: Não pode ser alterado o texto deste ficheiro fora deste anexo.

Problema 1

Votos desperdiçados:

$$waste = \bot$$

Corpo do ciclo-**for**:

$$step = \bot$$

Problema 2

Genes de $mergek$:

$$f = \bot$$
$$g = \bot$$

Extensão de $mSort$:

$$mSortk\ k = \bot$$

Problema 3

A detalhar na segunda edição deste enunciado

Problema 4

A detalhar na segunda edição deste enunciado

Index

References

[1] D.E. Knuth. *Literate Programming*. CSLI Lecture Notes Number 27. Stanford University Center for the Study of Language and Information, Stanford, CA, USA, 1992.

[2] J.N. Oliveira. *Program Design by Calculation*, 2018. Draft of textbook in preparation. viii+297 pages. Informatics Department, University of Minho.